# caffe paython 环境配置

> export PYTHONPATH=$HOME/Downloads/caffe/python:$PYTHONPATH

# caffe 安装

> git clone https://github.com/BVLC/caffe.git

安装依赖

```
sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev
libopencv-dev libhdf5-serial-dev protobuf-compiler
sudo apt-get install --no-install-recommends libboost-all-dev

sudo apt-get install libopenblas-dev

sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev
```

Makefile.config

```
## Refer to http://caffe.berkeleyvision.org/installation.html
# Contributions simplifying and improving our build system are welcome!

# cuDNN acceleration switch (uncomment to build with cuDNN).
# USE_CUDNN := 1

# CPU-only switch (uncomment to build without GPU support).
# CPU_ONLY := 1

# uncomment to disable IO dependencies and corresponding data layers
# USE_OPENCV := 0
# USE_LEVELDB := 0
# USE_LMDB := 0

# uncomment to allow MDB_NOLOCK when reading LMDB files (only if necessary)
#   You should not set this flag if you will be reading LMDBs with any
#   possibility of simultaneous read and write
# ALLOW_LMDB_NOLOCK := 1

# Uncomment if you're using OpenCV 3
 OPENCV_VERSION := 3

# To customize your choice of compiler, uncomment and set the following.
# N.B. the default for Linux is g++ and the default for OSX is clang++
# CUSTOM_CXX := g++

# CUDA directory contains bin/ and lib/ directories that we need.
CUDA_DIR := /usr/local/cuda
# On Ubuntu 14.04, if cuda tools are installed via
# "sudo apt-get install nvidia-cuda-toolkit" then use this instead:
# CUDA_DIR := /usr

# CUDA architecture setting: going with all of them.
# For CUDA < 6.0, comment the lines after *_35 for compatibility.
CUDA_ARCH := -gencode arch=compute_30,code=sm_30 \
             -gencode arch=compute_35,code=sm_35 \
             -gencode arch=compute_50,code=sm_50 \
             -gencode arch=compute_52,code=sm_52 \
             -gencode arch=compute_61,code=sm_61 \
             -gencode arch=compute_61,code=compute_61

             # -gencode arch=compute_20,code=sm_20 \
             # -gencode arch=compute_20,code=sm_21 \



# BLAS choice:
# atlas for ATLAS (default)
# mkl for MKL
# open for OpenBlas
# BLAS := atlas
BLAS := open
# Custom (MKL/ATLAS/OpenBLAS) include and lib directories.
# Leave commented to accept the defaults for your choice of BLAS
# (which should work)!
# BLAS_INCLUDE := /path/to/your/blas
# BLAS_LIB := /path/to/your/blas

# Homebrew puts openblas in a directory that is not on the standard search
path
# BLAS_INCLUDE := $(shell brew --prefix openblas)/include
# BLAS_LIB := $(shell brew --prefix openblas)/lib

# This is required only if you will compile the matlab interface.
# MATLAB directory should contain the mex binary in /bin.
# MATLAB_DIR := /usr/local
# MATLAB_DIR := /Applications/MATLAB_R2012b.app

# NOTE: this is required only if you will compile the python interface.
# We need to be able to find Python.h and numpy/arrayobject.h.
PYTHON_INCLUDE := /usr/include/python2.7 \
        /usr/lib/python2.7/dist-packages/numpy/core/include
# Anaconda Python distribution is quite popular. Include path:
# Verify anaconda location, sometimes it's in root.
# ANACONDA_HOME := $(HOME)/anaconda2
# PYTHON_INCLUDE := $(ANACONDA_HOME)/include \
        $(ANACONDA_HOME)/include/python2.7 \
        $(ANACONDA_HOME)/lib/python2.7/site-packages/numpy/core/include \

# Uncomment to use Python 3 (default is Python 2)
# PYTHON_LIBRARIES := boost_python3 python3.5m
# PYTHON_INCLUDE := /usr/include/python3.5m \
#                 /usr/lib/python3.5/dist-packages/numpy/core/include
# We need to be able to find libpythonX.X.so or .dylib.
PYTHON_LIB := /usr/lib
# PYTHON_LIB := $(ANACONDA_HOME)/lib

# Homebrew installs numpy in a non standard path (keg only)
# PYTHON_INCLUDE += $(dir $(shell python -c 'import numpy.core;
print(numpy.core.__file__)'))/include
# PYTHON_LIB += $(shell brew --prefix numpy)/lib

# Uncomment to support layers written in Python (will link against Python
libs)
# WITH_PYTHON_LAYER := 1

# Whatever else you find you need goes here.
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
/usr/include/hdf5/serial/
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib  /usr/lib/x86_64-
linux-gnu/hdf5/serial

# If Homebrew is installed at a non standard location (for example your
home directory) and you use it for general dependencies
# INCLUDE_DIRS += $(shell brew --prefix)/include
# LIBRARY_DIRS += $(shell brew --prefix)/lib

# Uncomment to use `pkg-config` to specify OpenCV library paths.
# (Usually not necessary -- OpenCV libraries are normally installed in one
of the above $LIBRARY_DIRS.)
# USE_PKG_CONFIG := 1

# N.B. both build and distribute dirs are cleared on `make clean`
BUILD_DIR := build
DISTRIBUTE_DIR := distribute

# Uncomment for debugging. Does not work on OSX due to
https://github.com/BVLC/caffe/issues/171
# DEBUG := 1

# The ID of the GPU that 'make runtest' will use to run unit tests.
TEST_GPUID := 0

# enable pretty build (comment to see full commands)
Q ?= @
```

安装

> sudo make -j8
> sudo make distribute

应用

CMakeLists.txt

```
set(CAFFE_PATH "$ENV{HOME}/Disk/caffe/distribute")
if (EXISTS "${CAFFE_PATH}")
    include_directories(
            include
            ${catkin_INCLUDE_DIRS}
            ${OpenCV_INCLUDE_DIRS}
            ${CAFFE_PATH}/include
    )
    target_link_libraries(lidar_cnn_seg_detect
            ${catkin_LIBRARIES}
            ${OpenCV_LIBRARIES}
            ${CUDA_LIBRARIES}
            ${CAFFE_PATH}/lib/libcaffe.so
            glog
            )
```

caffe python layer

layer tuozhan

The models are defined in plaintext protocol buffer schema (prototxt) while the learned models are serialized as binary protocol buffer (binaryproto) .caffemodel files.

> caffe 训练需要数据集，网络.prototxt 和 solver.prototxt

caffe 网络可视化

> python draw_net.py --rankdir TB test.prototxt test.png